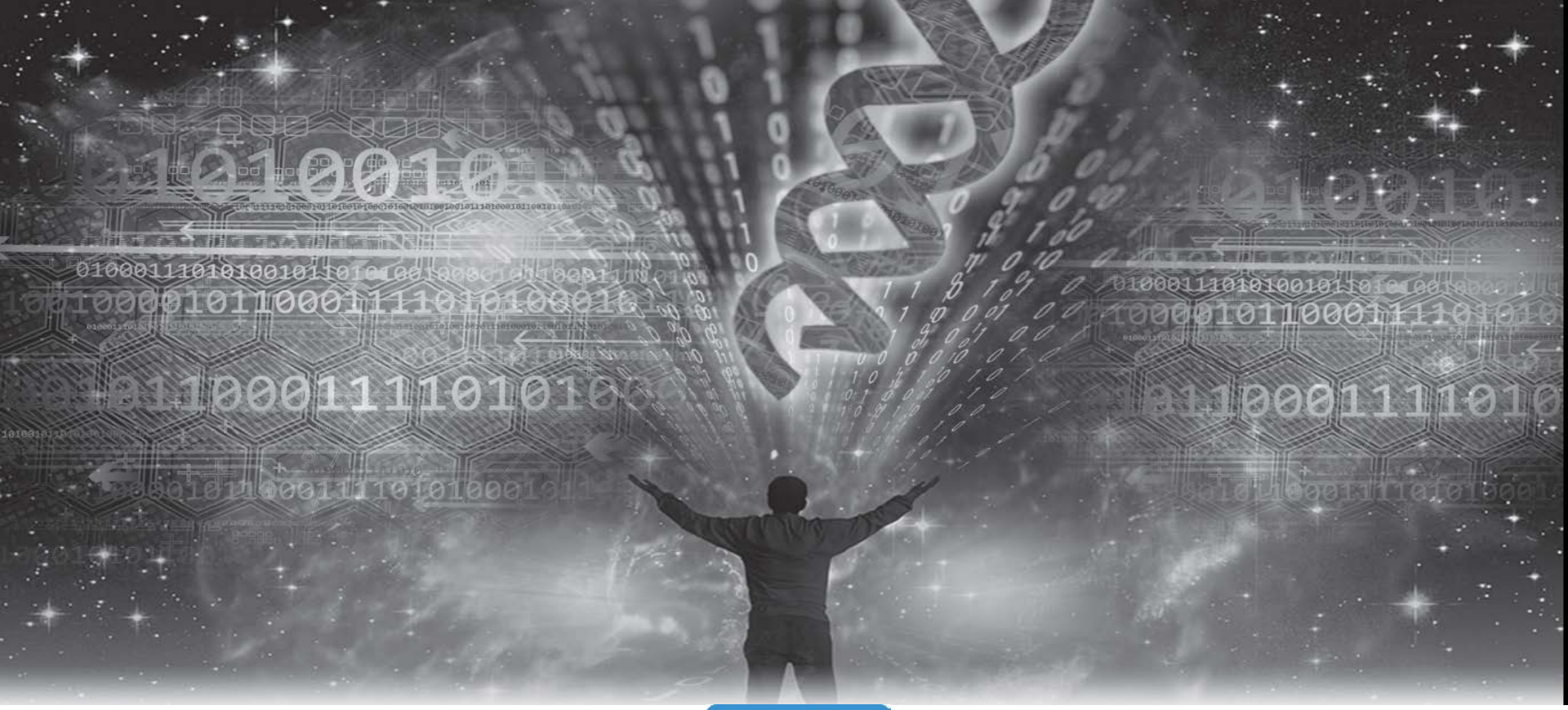
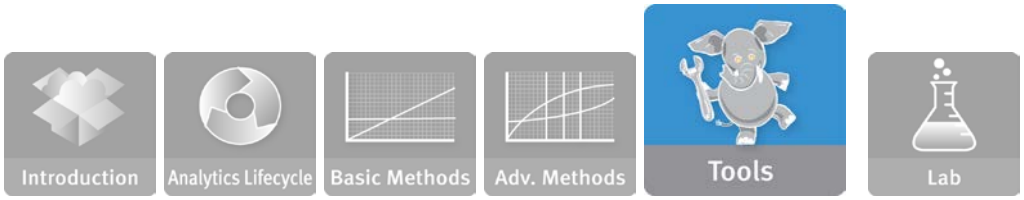


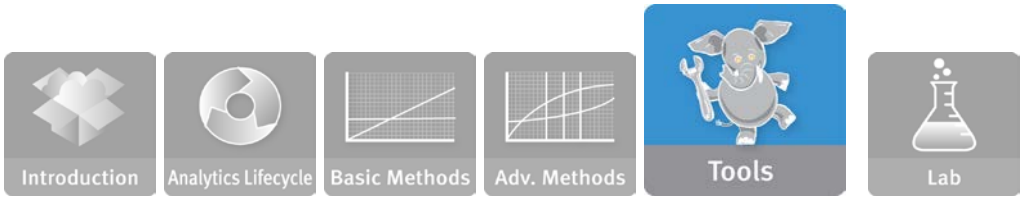
Module 5 – Advanced Analytics - Technology and Tools



Module 5: Advanced Analytics - Technology and Tools

Upon completion of this module, you should be able to:

- Perform Analytics on Unstructured data using MapReduce Programming paradigm
- Use Hadoop, HDFS, HIVE,PIG and other products in the Hadoop ecosystem for unstructured data analytics
- Effectively use advanced SQL functions and Greenplum extensions for in-database analytics
- Use MADlib to solve analytics problems in-database



Module 5: Advanced Analytics - Technology and Tools

Lesson 1: Analytics for Unstructured Data - MapReduce and Hadoop

During this lesson the following topics are covered:

- MapReduce & Hadoop
- HDFS – the Hadoop Distributed File System
- YARN – Yet Another Resource Negotiator

Putting the Data Analytics Lifecycle into Practice

- MapReduce & Hadoop in Data Analytics Life Cycle
  - ✓Phase 1: Discovery
  - ✓Phase 2: Data Preparation
  - Phase 3: Model Planning
  - ✓Phase 4: Model Building
  - Phase 5: Communicate results
  - Phase 6: Operationalize
- You have “big data,” how can you make it suitable for analysis?
  - That is, obtain results & key findings in a timely manner?
- MapReduce represents activities in Phase 1 and 2 of the life cycle in acquire/parse and filter stages and also in Phase 4 with model buliding

Big Data is the Next Big Thing

Online Search

Google

2,000,000+ searches per **minute**

Geo Sensor

foursquare

Social Media

Youtube

72 hours of video are uploaded every **minute**

Mobile Game

Social Network

Twitter

130,000+ tweets per **minute**

Online Review

yelp

tripadvisor

Why MapReduce?

"In Pioneer days,
they used oxen for heavy pulling.
When one ox couldn't budge a log, they didn't try to grow a larger ox…

We shouldn't be trying to grow bigger computers,
but to add more systems of computers."
Grace Hopper

The MapReduce paradigm helps you add more oxen

By definition, big data is too large to handle by conventional means. Sooner or later, you just can't scale up anymore

What is MapReduce?

- A parallel programming model suitable for big data processing
  - Split data into distributable chunks ("shards")
  - Define the steps to process those chunks
  - Run that process in parallel on the chunks

- Scalable by adding more machines to process chunks
  - Leverage commodity hardware to tackle big jobs

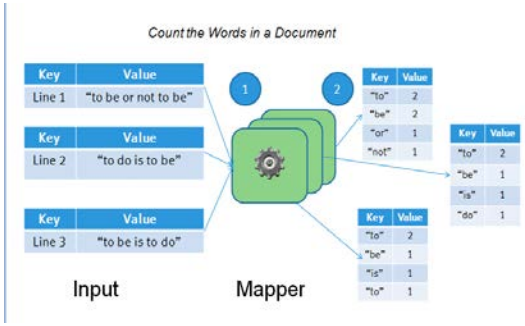
- The foundation for Hadoop
  - **MapReduce** is a **parallel programming model**
  - **Hadoop** is a **concrete platform that implements MapReduce**

When to Use MapReduce

- Problems that are “embarrassingly parallel”
- Examples
  - Word count
  - Reverse index
  - tf-idf
  - Distributed *grep* and distributed object recognition ("Where's Waldo?")
  - Distributed "associative" aggregation (marginalization, sum; mean if you track both numerator and denominator; min or max; count)
    - Hadoop calls them "combiners"

The Map part of MapReduce

- Transform
  - (Map) input values to output values: <k1,v1> → <k2,v2>
- Input – Key/Value Pairs
  - For instance, Key = line number, Value = text string
- Map Function
  - Steps to transform input pairs to output pairs
    - For example, count the different words in the input
- Output – Key/Value Pairs
  - For example, Key = <word>, Value = <count>
- Map output is the input to Reduce

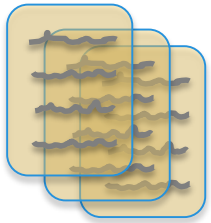


The Reduce Part of MapReduce

- Merge (Reduce) Values from the Map phase
  - Reduce is optional. Sometimes all the work is done in the Mapper
- Input
  - Values for a given Key from all the Mappers
- Reduce Function
  - Steps to combine (Sum?, Count?, Print?,…) the values
- Output
  - Print values?, load into a DB? send to the next MapReduce job?

Motivating Example: Word Count

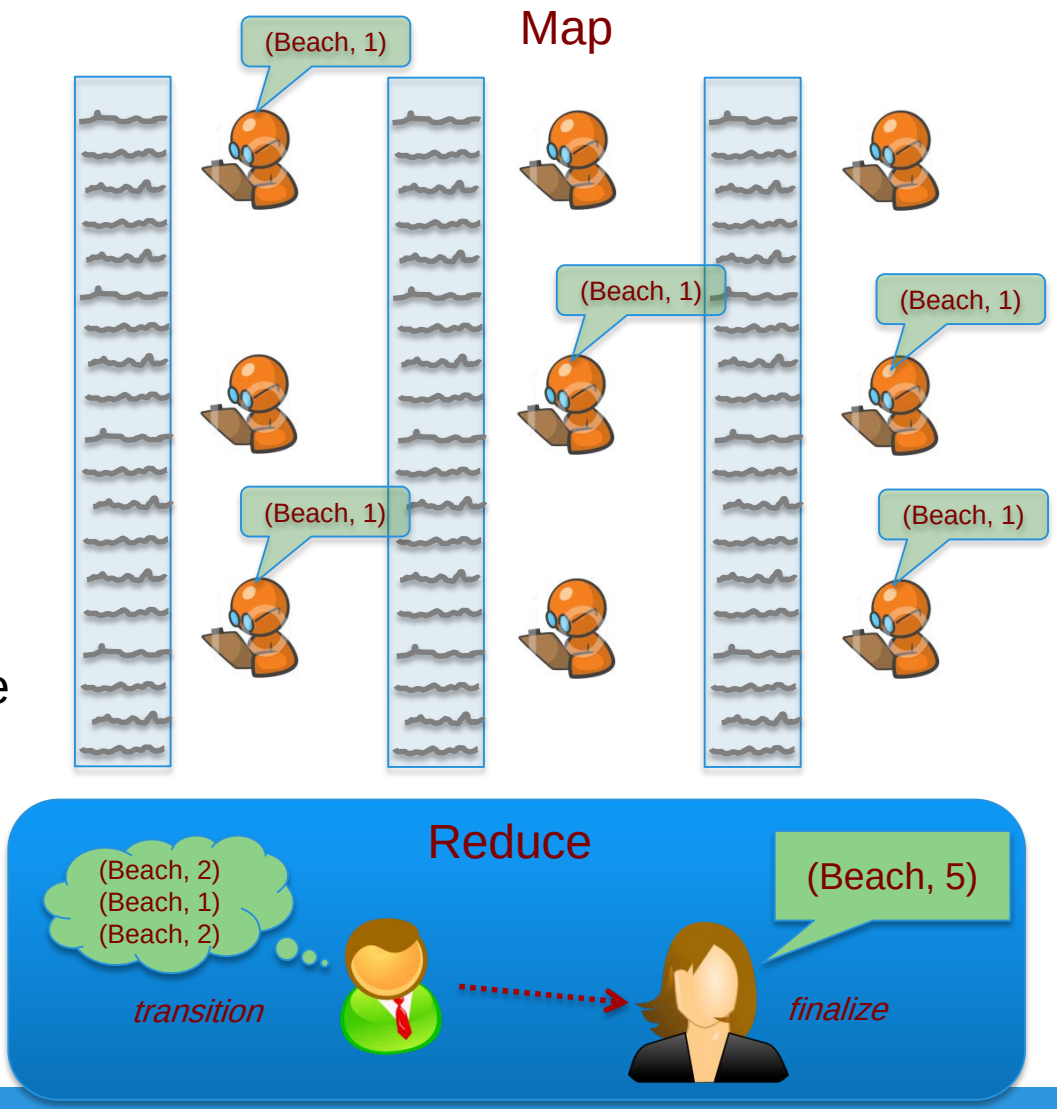
This is the “Hello World” of MapReduce



Distribute the text of millions of documents over hundreds of machines.

MAPPERS can be word-specific. They run through the stacks and shout “One!” every time they see the word “beach”

REDUCERS listen to all the Mappers and total the counts for each word.



Example: Social Triangles (e-discovery)

Suppose you have 517,424 emails from an energy company under indictment.

The social network may be implied by all To:From pairings in the emails, with reflexivity accounted for.

Date: Thu, 4 May 2000 09:55:00 -0700 (PDT)
From: walt.zimmerman@enron.com
To: michael.burke@enron.com, dana.gibbs@enron.com, lori.maddox@enron.com, susan.ralph@enron.com
Subject: Update on Steve Todoroff Prosecution--CONFIDENTIAL/SUBJECT TO ATTORNEY-CLIENT PRIVILEGE
Cc: steve.duffy@enron.com, stanley.horton@enron.com, jdegeeter@velaw.com

Almost one month ago, Special Agent Carl Wake of the FBI called me about the Steve Todoroff investigation. He indicated that the FBI had recently learned of the article about EOTT's NGL theft that appeared in the business section of the Houston Chronicle. Mr. Wake said it might be a matter the FBI would like to investigate. I told Mr. Wake that EOTT was currently working with the Harris County District Attorney on the prosecution of this matter, and I thanked him for the FBI's interest. He told me that the FBI might want to work with the Harris County District Attorney in investigating this matter, and he stated that there may be investigative information that the FBI can obtain more quickly than the Harris County District Attorney. Mr. Wake requested a copy of the materials we had provided to the Harris County District Attorney.

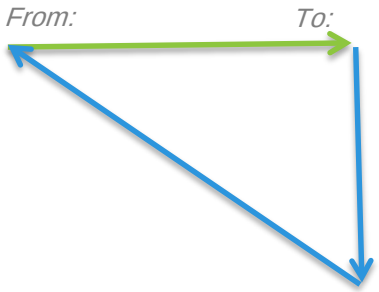
In order to avoid damage to the good rapport we have established with Assistant District Attorney Bill Moore, I asked John DeGeeter to call Bill Moore and advise him of the contact that had been made by the FBI. Bill Moore agreed to call Carl Wake and work with Mr. Wake on his request for the materials provided by EOTT.

Carl Wake called me again yesterday. He has been working with Bill Moore. Mr. Wake stated it was too early to speculate as to what charges would be brought. He did say that our materials clearly indicated federal wire fraud and possibly mail fraud. He said that where there is wire fraud, there is usually money laundering.

The purpose of Mr. Wake's call yesterday was to inquire about the status of some interview summaries that John DeGeeter and I have prepared and collected at the request of Bill Moore. Mr. Wake requested that EOTT send a copy of the summaries to him when we sent the summaries to Bill Moore. Those summaries were sent out today.

I gathered from my calls with Carl Wake that the FBI is very interested in taking an active part in this investigation. In order to build on the relationship we have established with Bill Moore, we will continue to direct our inquires about the investigation to Mr. Moore until he tells us to do otherwise.

Social Triangle: First Directed Edge



Mapper1

Maps two regular expression searches:
To: Michael, Dan, Lori, Susan
From: Walt

Emits the outbound directed edge of the social graph:
<Key, Value> = <Walt, [Michael, Dan, Lori, Susan]>

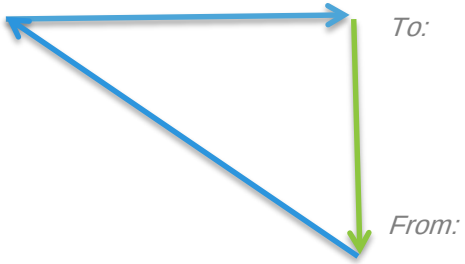
Reducer1

Gets the output from the mapper with different values
<Key, Value> = <Walt, [Michael, Dan, Lori, Susan]>
<Key, Value> = <Walt, [Lori, Susan, Jeff, Ken]>

Unions the values for the second directed edge:
<Key, Value> = <Walt, [Dan, Jeff, Ken, Lori, Michael, Susan]>

Social Triangle: Second Directed Edge

Mapper2



Reverses the previous Map:
To: Michael, Dan, Lori, Susan
From: Walt
Emits the inbound directed edge of the social graph:
<Key, Value> = <Susan, Walt>; <Lori, Walt>; <Dan, Walt>; etc

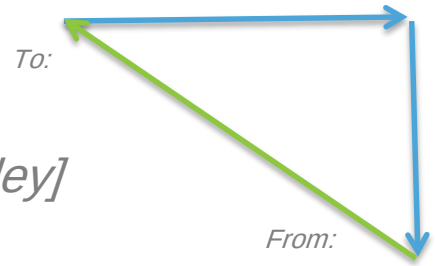
Reducer2

Gets the output from the mapper with different values
<Key, Value> = <Susan, Walt>
<Key, Value> = <Susan, Jeff>

Unions the values for the third directed edge:
<Key, Value> = <Susan, [Jeff, Ken, Walt]>

Social Triangle: Third Directed Edge

Mapper3



Join [inbound] and [outbound] lists by Key
Walt, [Jeff, Ken, Lori, Susan], [Jeff, Lori, Stanley]

Emits <Person, Person> pair with level of association:
<Key, Value> = <Walt: Jeff reciprocal>; <Walt:Stanley directed>, etc

Reducer3

Reducer unions the output of the mappers and presents rules:
<Key, Value> = <Walt::Jeff, reciprocal>
<Key, Value> <Walt::Stanley, directed>

The third reducer can shape the data any way that serves the business objective.

What is .....  ?

People use “Hadoop” to mean one of four things:

- MapReduce paradigm.
- Massive unstructured data storage on commodity hardware.

(ideas)

- Java Classes for HDFS types and MapReduce job management.
- HDFS: The Hadoop distributed file system.

(actual Hadoop)

With Hadoop, you can do MapReduce jobs quickly and efficiently.

What do we Mean by Hadoop

- A framework for performing big data analytics
  - An implementation of the MapReduce paradigm
  - Hadoop glues the storage and analytics together and provides reliability, scalability, and management

Two Main Components

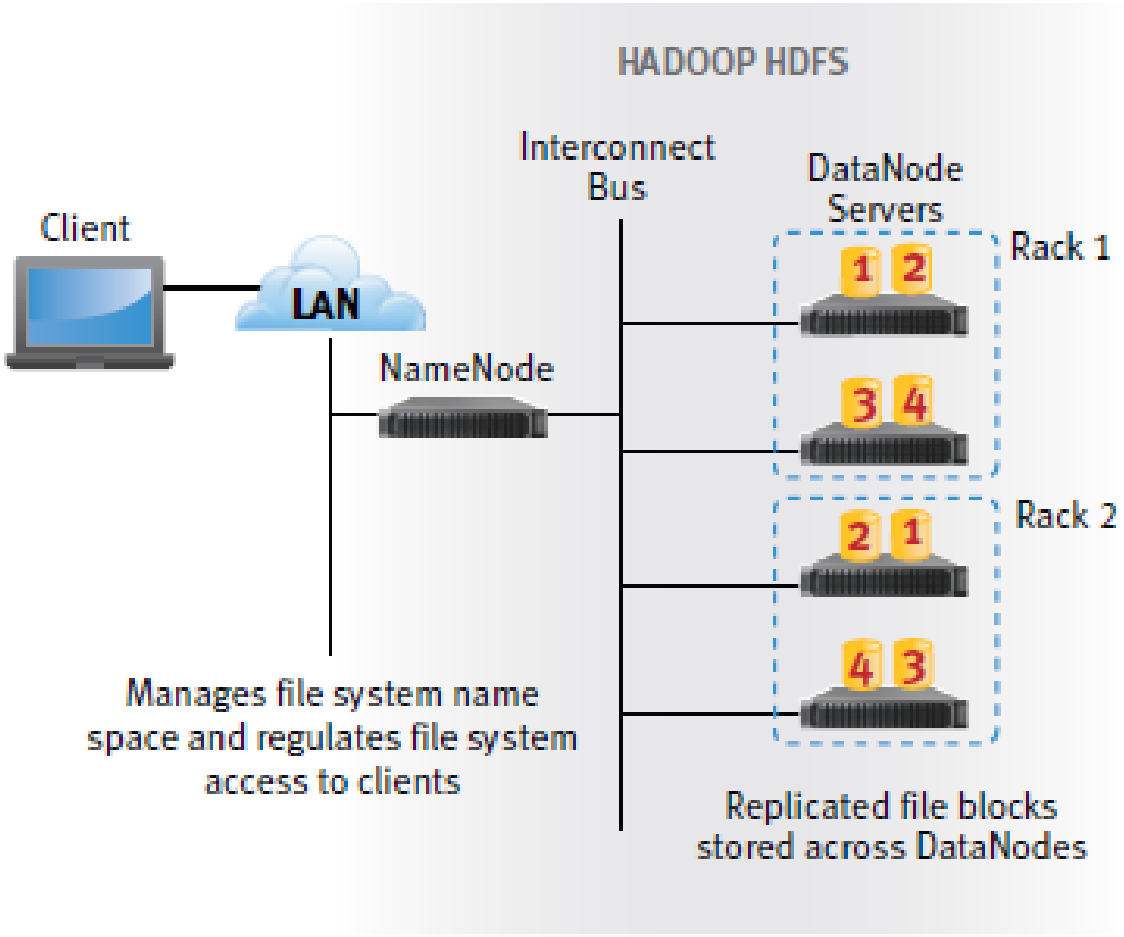
Storage (Big Data)

- HDFS – Hadoop Distributed File System
- Reliable, redundant, distributed file system optimized for large files

MapReduce (Analytics)

- Programming model for processing sets of data
- Mapping inputs to outputs and reducing the output of multiple Mappers to one (or a few) answer(s)

Hadoop and HDFS



Hadoop Operational Modes

- Java MapReduce Mode
  - Write Mapper, Combiner, Reducer functions in Java using Hadoop Java APIs
  - Read records one at a time
- Streaming Mode
  - Uses *nix pipes and standard input and output streams
  - Any language (Python, Ruby, C, Perl, Tcl/Tk, etc.)
  - Input can be a line at a time, or a stream at a time

Example: Hadoop Streaming Mode

	Script to invoke Hadoop
1	`Hadoop jar $HADOOP_INSTALL/contrib/streaming/hadoop-*-streaming.jar \`
2	`-input input/ \ # relative to HDFS`
3	`-output output \ # relative to HDFS`
4	`-mapper mapper.py \`
5	`-reducer reducer.py \`
6	`-file scripts/mapper.py \`
7	`-file scripts/reducer.py`

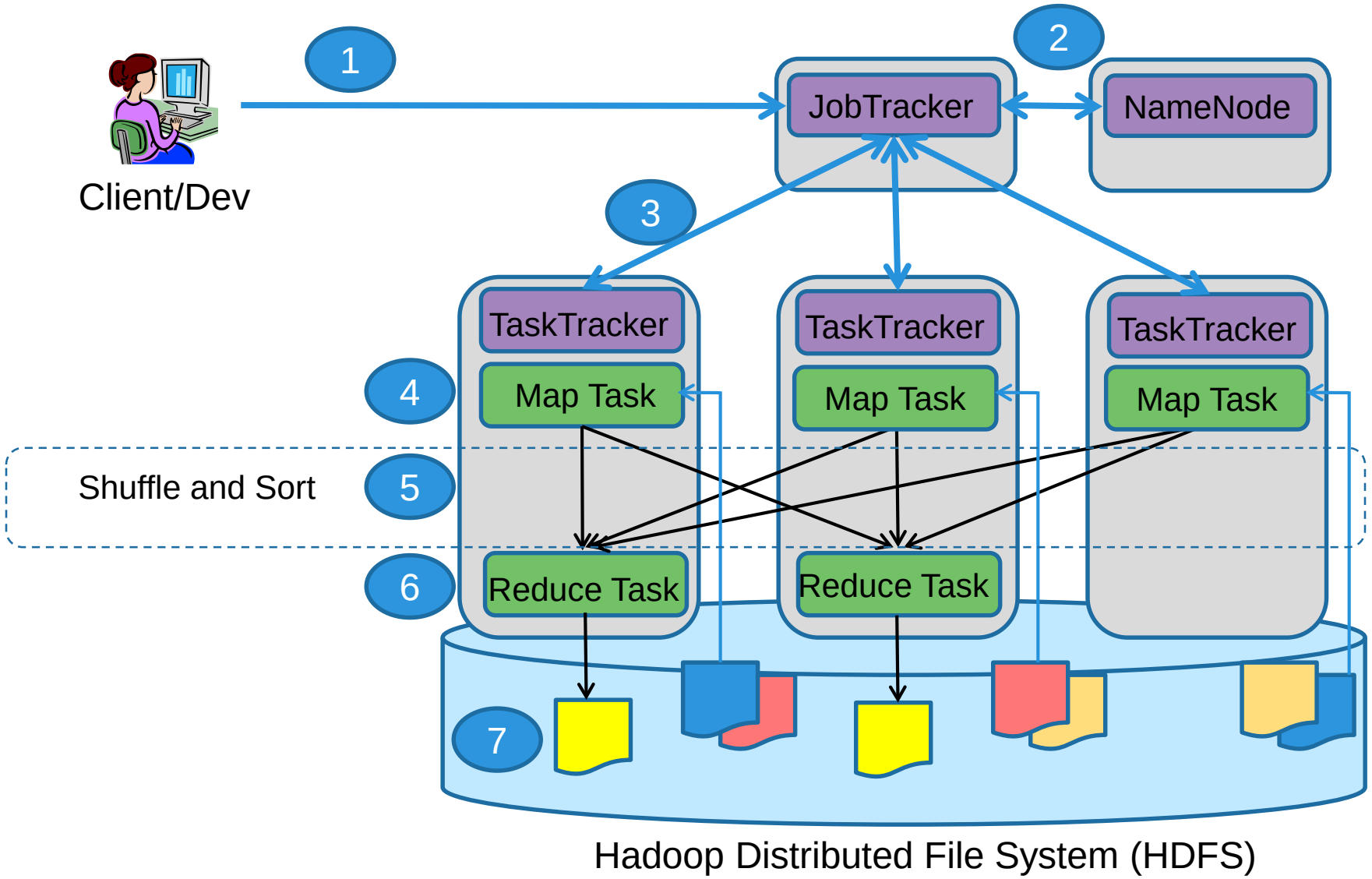
Simple Example of a Mapper: mapper.py

Code	Comments
import sys	
for line in sys.stdin	# input comes from STDIN
line = line.strip()	# strip leading/trailing whitespace
words = line.split()	# split line based on whitespace
for word in words: print '%s\t %s %(word, 1)	# for each word in the collection words # write word and count of "1", tab delimited

Simple Example of a Reducer: reducer.py

Code	Code, continued
cur_count = 0	if cur_word == word:
cur_word = None	cur_count += count
for line in sys.stdin	else:
line = line.strip()	if cur_word:
word, count = line.split("\t",1)	print '%s\t%s' (cur_word, cur_count)
try	cur_count, cur_word = (count, word)
count = int(count)	if cur_word == word:
except ValueError:	print '%s\t%s' %(cur_word, cur_count)
continue	

Putting it all Together: MapReduce and HDFS

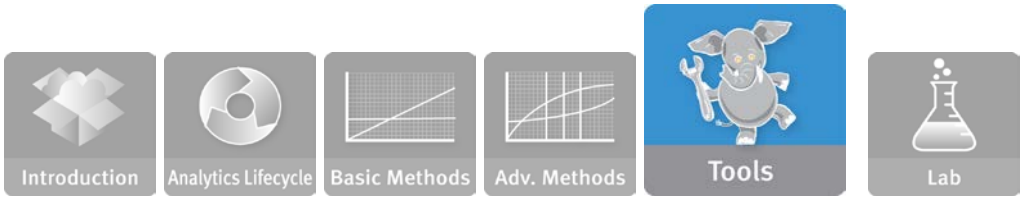


Using R with Hadoop

- The brute force way
  - Reading
    - rcon <- pipe("hadoop fs –cat output/d99/*", open="r")
    - readLines(rcon)
    - close(rcon)
  - Writing
    - wcon <- pipe("hadoop fs –put – dir1/dir2/out.txt", open="w")
    - cat(…, file=wcon)
    - close(wcon)
  - MapReduce
    - system("MapReduceJob.sh", wait=true)
    - # blocks until done

Using RHadoop

- 3 packages courtesy of Revolution R
  - rhdfs -- access to hdfs functions
  - rhbase – access to HBase
  - rmr – run MapReduce job
- HDFS
  - `hdfs.read(), …`
- HBase
  - `hb.list.tables(), …`
- MapReduce
  - `mapreduce(input_dir, output_dir, RMapFunction, RReduceFunction, … )`

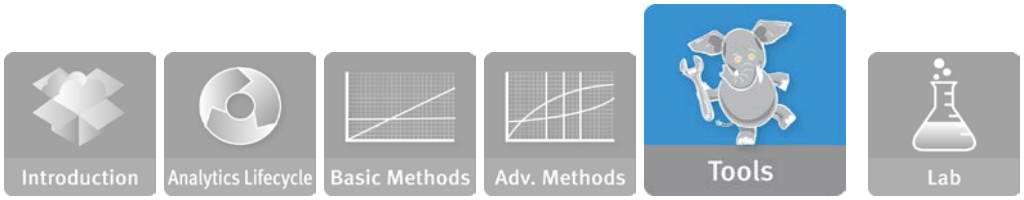


Module 5: Advanced Analytics - Technology and Tools

Lesson 1: Summary

During this lesson the following topics were covered:

- Applying the MapReduce/Hadoop Processing Framework in big data analytics problems for un-structured data
- Differentiating among the various elements of Hadoop
- Naming the components of HDFS
- Identifying the parts of an Hadoop streaming script



Module 5: Advanced Analytics – Technology and Tools

Lesson 2: The Hadoop Ecosystem

During this lesson the following topics are covered:

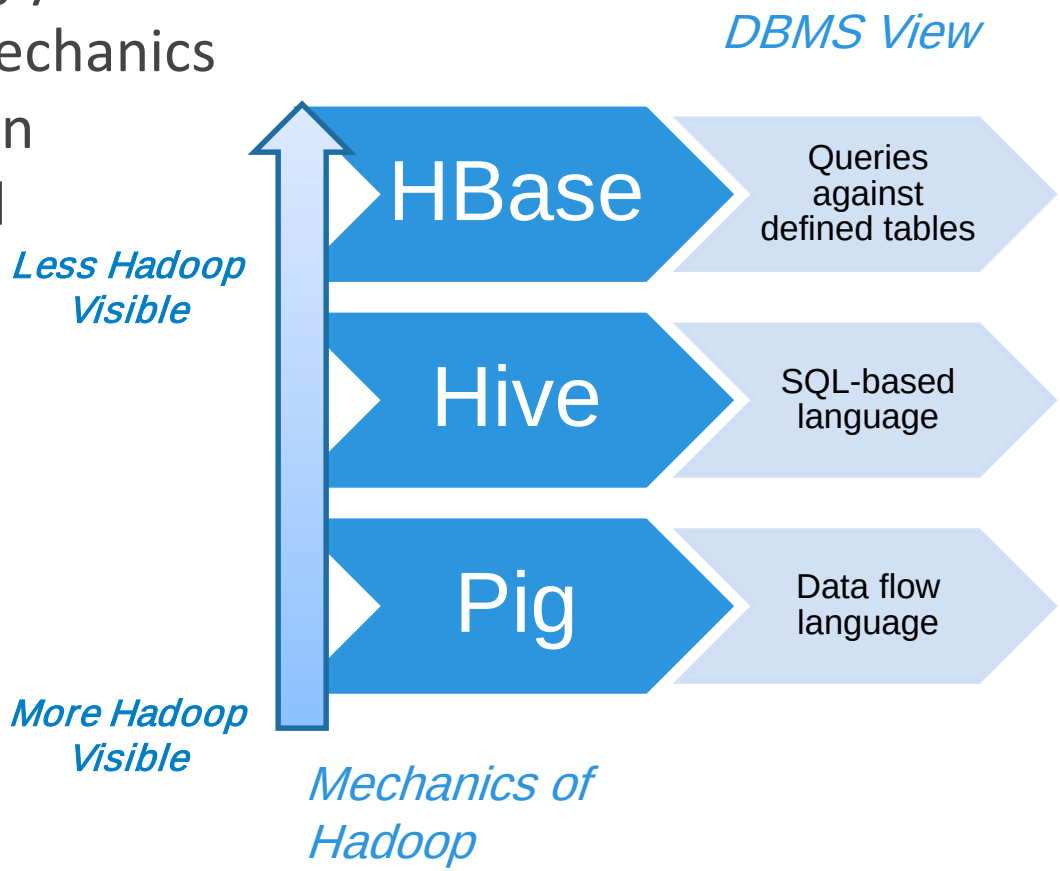
- Invoke interfaces from the command line
- Use query languages (Hive and Pig) for data analytics problems using un-structured data
- Build and query an HBase database
- Suggest examples where HBase is most suitable

Query Languages for Hadoop

- Builds on core Hadoop (MapReduce and HDFS) to enhance the development and manipulation of Hadoop clusters
  - **Pig** --- Data flow language and execution environment
  - **Hive** (and HiveQL) --- Query language based on SQL for building MapReduce jobs
  - **HBase** --- Column oriented database built on HDFS supporting MapReduce and point queries
    - **Depends on Zookeeper -** a coordination service for building distributed applications

Levels of Abstraction

- As you move from Pig to Hive to HBase, you are increasingly moving away from the mechanics of Hadoop and creating an RDBMS view of the world



What is Pig?

- Data flow language and execution environment for Hadoop

Two Main Elements

- A **data flow language** (Pig Latin)
- Two modes execution environment:
  - Local --- access a local file system
  - MapReduce when you're interested in the Hadoop environment
- When NOT to use Pig
  - If you only want to touch a small portion of the dataset (Pig eats it all)
  - If you do NOT want to use batch processing
    - Pig ONLY supports batch processing

Writing Pig Latin. Seriously.

- A Pig script is a series of operations (transformations) applied to an input to produce an output
  - May be helpful to think of a Unix pipe command
    - `tr [A-Za-z] file1; sort –o file2 file1; uniq –c file2`
- Supports examining data structures and subsets of data
- Can execute Pig programs as a script
  - Via Grunt - an interactive shell or from a Java program
  - Via the command line: `pig <scriptname>`

Deconstructing Pig

	-- max_temp.pig -- Finds the max temperature by year
1	`records = LOAD 'data/samples.txt'` `AS (year:chararray, temperature:int, quality:int) ;`

2	`filtered_records = FILTER records BY temperature != 9999`
	`AND (quality == 0 OR quality ==1 OR quality == 4 OR`
	`quality == 5 OR quality == 9);`

3	`grouped_records = GROUP filtered_records BY year;`
4	`max_temp = FOREACH grouped_records GENERATE group,` `MAX(filtered_records.temperature) ;`
5	`DUMP max_temp ;`

Pig Comparison with a SQL

Pig	SQL
A data flow language	A declarative programming language
Schema is optional, can be specified at run-time	Schema is required at data load time
Supports complex, nested data structures	Typically uses simple table structures
Does not support random reads or queries	Random reads and queries are supported

Hive and HiveQL

- Query language based on SQL for building MapReduce jobs
- All data stored in tables; schema is managed by Hive
  - Schema can be applied to existing data in HDFS

Hive Shell and HiveQL

- Hive
  - Provides web, server and shell interfaces for clients
  - Hive shell is the default
    - Can run external host commands using “!prog” command
    - Can access HDFS using the DFS command
- HiveQL
  - Partial implementation of SQL-92 (closer to MySQL)
  - Data in Hive can be in internal tables or “external” tables
    - Internal tables managed by Hive
    - External tables are not (lazy create and load)

Temperature Example: Hive

	Example Hive Code
1	`CREATE TABLE records (year STRING, temperature INT, quality INT) ROW FORMAT DELIMITED FIELDS TERMINATED by '\t' ;`
2	`LOAD DATA LOCAL 'data/samples.txt' OVERWRITE INTO TABLE records ;`
3	`SELECT year, MAX(temperature) FROM records WHERE temperature != 9999 AND (quality == 0 OR quality == 1 OR quality == 4 OR quality == 5 OR quality == 9)  GROUP BY year;`

Hive Comparison with a SQL

Hive	Database
“Schema on Read”	“Schema on Write”
Incomplete SQL-92 (never a design goal)	Full SQL-92
No updates, transactions. Indexes available in v0. 7	Updates, transactions and indexes.

APACHE HBASE - the Hadoop Database

- "Column oriented" database built over HDFS supporting MapReduce and point queries
- Depends on Zookeeper for consistency and Hadoop for distributed data.
- The Siteserver component provides several interfaces to Web clients (REST via HTTP, Thrift and Avro)

When to Choose HBase

- You need random, real-time read/write access to your big data
- You need sparse tables consisting of millions of rows and millions of columns where each column variable may be versioned
- Google's BigTable: a "Web table"

HBase Comparison with a Traditional Database

HBase	DBMS
No real indexes	Real indexes
Support automatic partitioning	No automatic partitioning
Ideal for billions of rows and millions of columns	Challenged by sparse data
Supports real time reads and writes	Supports real time reads and writes

Which Interface Should You Choose?

Pig

- Replacement for MapReduce Java coding
- When need exists to customize part of the processing phases (UDF)

Hive

- Use when SQL skills are available
- Customize part of the processing via UDFs

HBase

- Use when random queries and partial processing is required, or when specific file layouts are needed

Mahout

- Scalable machine learning and data mining library for Hadoop
- Support for four use cases
  - Recommendation mining
  - Classification
  - Clustering
  - Frequent itemset
- Requires Hadoop infrastructure and Java programming

Algorithms Available in Mahout

- Recommenders
  - Non-distributed recommenders
  - Distributed item-based collaborative filtering
  - Collaborative filtering using a parallel matrix classification
- Classification
  - Logistic Regression
  - Bayesian
  - Random Forests
  - Restricted Boltzmann Machines
  - Online Passive Aggressive
- Frequent Itemset
  - Parallel FP Growth Mining
- Clustering
  - Canopy Clustering
  - K-Means Clustering
  - Fuzzy K-Means
  - Mean Shift Clustering
  - Hierarchical Clustering
  - Dirichlet Process Clustering
  - Latent Dirichlet Allocation
  - Spectral Clustering
  - Minhash Clustering